COMPONENTWARE APPROACHES IN MANAGEMENT INFORMATON SYSTEMS

Dipl.-Ing. Jürgen Kaster, Dipl.-Ing. Annette Kaster
Ergonomics and Information Systems Department (EFS)
Research Institute for Communication, Information Processing, and Ergonomics (FKIE)
Research Establishment for Applied Natural Sciences (FGAN)
Neuenahrer Str. 20, D-53343 Wachtberg-Werthhoven, Germany
Tel: [+49] (228) 9435 - 380, Fax: - 508
j.kaster@fgan.de, a.kaster@fgan.de

Summary

Modern command and control information systems (CCIS) are characterized by continuously changing conditions regarding technology, task and user profiles. As a consequence of this heterogeneity a huge amount of information and knowledge pieces of different data types has to be managed and processed in distributed communication networks. The situation in military CCIS is even more complex, regarding e.g. new requirements and multi-national command structures in actual out-of-area missions.

The paper will focus on architecture models on the basis of "*componentware technology*". Pursuing the proposed ideas may help to design systems of high flexibility that can be adapted to actual user needs and task requirements.

1 Challenges in the design of CCIS

Military CCIS are developed to solve operational requirements in the areas of situation identification and assessment, planning, decision making, and commanding. Appropriate timely information flow has to be provided over large, distributed communication networks. However, problems in actual command processes are not mainly induced by the complexity of military structures. There are three major influence coefficients that induce that those systems are unstable over time. These influence coefficients are rapidly changing conditions regarding applied technology, operational requirements, and heterogeneous user profiles.

In most cases it is not possible to design military information systems from scratch that have an overall consistent hardware and software basis for the technical and conceptual implementation. Operational requirements cause continuous updates and adaptations. As a consequence a rather *heterogeneous environment* evolves in which command and control processes must be embedded. Of course, this complicates the realization of an universal support concept and induces interruptions of the required communication flow. Information may be falsified or even lost.

Another problem derives from the fact that operational *mission requirements* have dramatically changed in the last few years. Out-of-area missions have to be planned, prepared and executed within a short time. Task forces have to be established in accordance to actual needs. This requires extremely high flexibility and adaptability of the underlying technical support systems. Additionally, multi-national missions and command structures are nowadays a matter of course.

A third problem area is the need of supporting total different *user profiles*. Users show discrepancies in their technical and professional competence, working styles, and they have different tasks to fulfill at the same system or system components.

The consequence is that the realization of universal and stable concepts in military CCIS is very difficult or even impossible. The well known problems, which are paraphrased by "software crisis", are established mainly by impediments in a *timely and task-oriented access* to information and knowledge that is stored somewhere in the system.

The challenge for improvements of CCIS from a human factors point of view is to provide direct problem-oriented access to the information that is actually needed in current operational situations. The next task is to make the human-machine interface as homogeneous as possible even in modern multimedia environments. The reason is that learning processes can be simplified if there is a high value of recognition implicitly in the applied dialogue procedures. However, one has to take into account that information items may change their characteristics during processing and evaluation. For example, a broad range of textual input details may be aggregated and transformed to other presentation categories to support decision processes at a higher command level.

To overcome restrictions and insufficiencies in military CCIS, it is necessary to make more efficient design concepts and processes available. Componentware technology is a new paradigm, that allows to construct flexible system architectures (macro-view) as well as concrete military application software (micro-view). The following chapter will focus on this technology.

Paper presented at the RTO HFM Symposium on "Usability of Information in Battle Management Operations", held in Oslo, Norway, 10-13 April 2000, and published in RTO MP-57.

2 Componentware – a new approach for CCIS development

2.1 Componentware as an extension of object-oriented techniques

The transfer from former procedural views to object-oriented software techniques introduced totally new ways of thinking and abstractions to the software engineering process. Object-oriented analysis (OOA) and object-oriented programming (OOP) emphasize concepts like e.g., abstraction, modularization, and reusability. However, object-orientation is limited to the reflection on technical structures like program and system objects and their interrelationships.

"As a system becomes more complex, the design problem goes beyond the algorithms and data structures of the computation: designing and specifying the overall system structures emerges as a new kind of problem" [Booch et. al. 1999]. In order to address this citation it is necessary to provide methods which represent a system as a whole assembled by independent but interacting components. *Frameworks* are general system models which structure the application in cooperating building blocks. Recently the *componentware* principle offers solutions to develop applications which are based on such frameworks (Schreiber-Ehle 1999). This technique defines standardized protocols which independently developed programs can use to communicate (Microsoft 1998; OMG 1998). If this technique is applied modules can be exchanged (even at runtime). Therefore, modifications of the system and its performance can be achieved even after the software installation process has been completed (Plasil et. al. 1998).

A software component is an executable piece of software that provides a useful, in the application context valuable functionality. It offers plug & play readiness for service and is cooperative in combination with other programs (Griffel 1998). The *component view* has its value in the emphasis of delimitations and independencies of software elements in supplementation to internal views on program and system objects. Components are independent application-oriented building blocks with well defined interfaces to other application software. Despite the required originality of a software component it becomes a "real" functional component (as a part of the whole) only in combination with others. This will result in a growing significance of modern framework techniques (as a model of the whole).

The component view may be applied to an entire information system as well as to special application software. This dualism is also important considering the design pattern which is introduced in the next chapter.

2.2 Model-View-Controller: Design pattern for improved data management

In order to create a suitable software architecture consisting of independent exchangeable parts the use of *design patterns* is helpful (Gamma et. al. 1995). Design patterns structure logical dependencies on a high abstraction level. They are applicable to a software design in general and they supply designers with ready-made design solutions. A useful architectural pattern to create componentware systems is the Model-View-Controller (MVC) paradigm.

An optimization of user interfaces as well as assistance systems according to ergonomic criteria is indispensable for an efficient support of complex command and control tasks with computer-based information systems. There have been benefits provided by the construction of separate user interface management systems (UIMS) in complex applications (Myers 1995). However, most of the applied software products are rigid monolithic blocks that cannot be adapted to current needs. The reason is that the internal data management is strongly tied to algorithms and computations.

The goal of the Model-View-Controller (MVC) design pattern is - in addition to the mentioned separation of the UIMS ("controller") – to separate the application object ("model") from the way it is represented to the user ("view"). This leads to a system structure as illustrated in fig. 1.

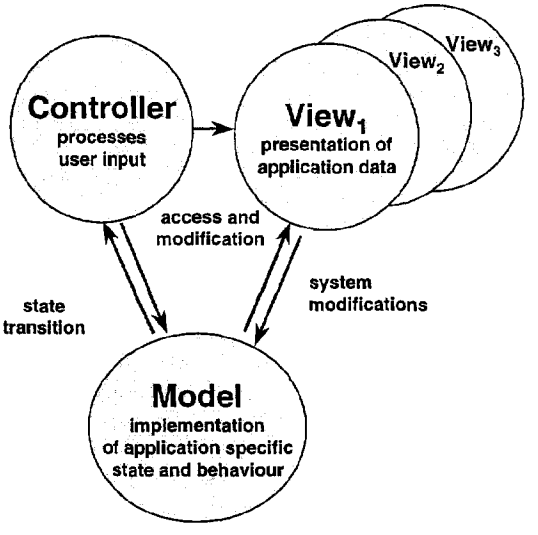

Figure 1: Model-View-Controller Design Pattern

Chapter 3.1 gives an example of a program that follows the MVC design concept. In a more general view, the MVC concept means, that in modern workstations which are normally used as front-end in CCIS, the data storage should be separated from the applied programs and tools. This can be done when powerful database management systems are available in a client server environment. When data storage and data processing capabilities are separated, user- and task-oriented systems can be developed that control the information flow between database and front-end tool.

This resulting flexibility in the construction of appropriate user interfaces will be demonstrated by a case study which is the topic of the next chapter.

3 Design of Human-Machine Interfaces based on Componentware Technology

The following case study demonstrates the benefits of the use of componentware technology which was introduced in chapter 2.

3.1 Scenario

In a former military conflict it was necessary to maintain a list of mine accidents which happened during the crisis. The task was to provide the appropriate information in an event list which was part of a situation report. This report was generated as a series of PowerPoint slides. Therefore it was decided to maintain the event list directly as a table in the presentation program.

Obviously, this way of "data managing" violated the MVC design pattern and comprised a lot of disadvantages. Problems occurred in the handling of the data (almost manual processing and formatting) and last but not least the data were not available for further use in the command and control process.

The current situation illustrated by this scenario emphasizes the necessity to conduct research on the scientific conception of improved task and user adaptable support systems based on systematic task and system analyses. Two examples will be presented below.

3.2 Application example (1)

The componentware approach is not only applicable to improve the architecture of a total CCIS (macro-view). It can also be used to construct concrete military application software (micro-view). The following is an example of such an improved military-off-the-shelf product (MOTS).

To study the impact of componentware architecture a military situation display system has been modified and implemented as a component. The complete human computer interface is now made up of commercial-off-the-shelf products (COTS), like word processor, presentation tool, image editor, and specialized elements, like situation editor (Kaster 1998) as well as geographic vector and raster map display and creation programs.

All these components can be put together at runtime according to the actual operational requirements. To achieve a smooth exchangeability of the specialized component elements the underlying software design is structured by strict separation of the user interface from the remaining system functionality. This separation means, e.g., that dialogs are implemented as independent components as well as menu bars. They do not belong to the kernel of the system. Therefore they can be easily replaced, even by third party components (e.g., a new color selection technique).

Furthermore well known design patterns (Gamma et. al. 1995) such as *Model-View-Controller* (MVC) paradigm, *Presentation-Abstraction-Controller* (PVC), and the *Factory* pattern have been used to group the parts logically. Because of the underlying structure of the MVC concept the user can choose between different means for presenting information, like graphics with or without geographic background, textual output of object structures and attributes (fig. 2), and for manipulating input data.

3.3 Application example (2)

Within the existing client/server infrastructure an alternative concept has been developed on the basis of a professional database management system. According to the requirements of the afore mentioned MVC pattern results from the analysis of the application domain were transformed into a database design (model) and an application design (view). While the underlying development process is described in chapter 4 the resulting application components and user interfaces are illustrated below.

Fig. 3 shows a standard input mask for mine incidents with sophisticated assistance for filling in the form, i.e., entering values for parameters and attributes describing an incident.. Fig. 4 contains several different views on the same data which are designed for the executive level of users. Furthermore, in the componentware environment it is possible to exchange data with other presentation tools (e.g., preformatted Excel spreadsheets, PowerPoint slideshows).

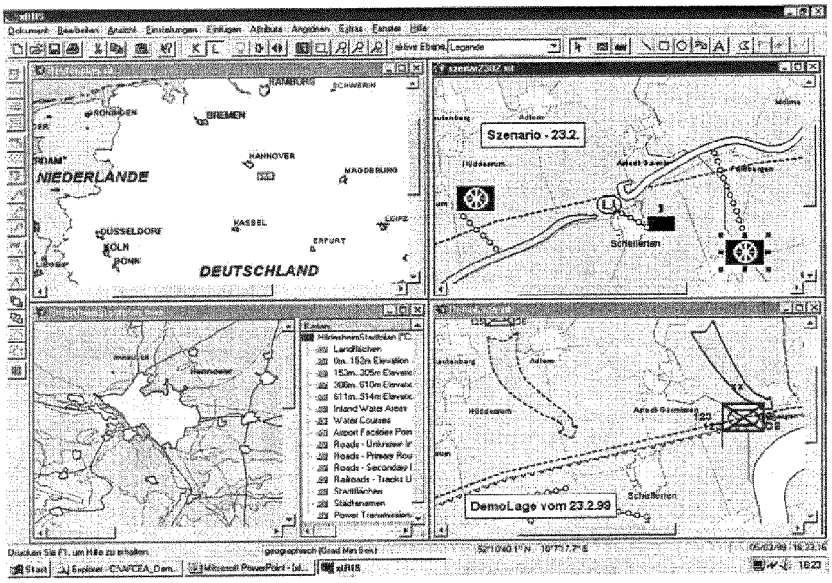

Figure 2: Situation data presented in a multiple windows environment.

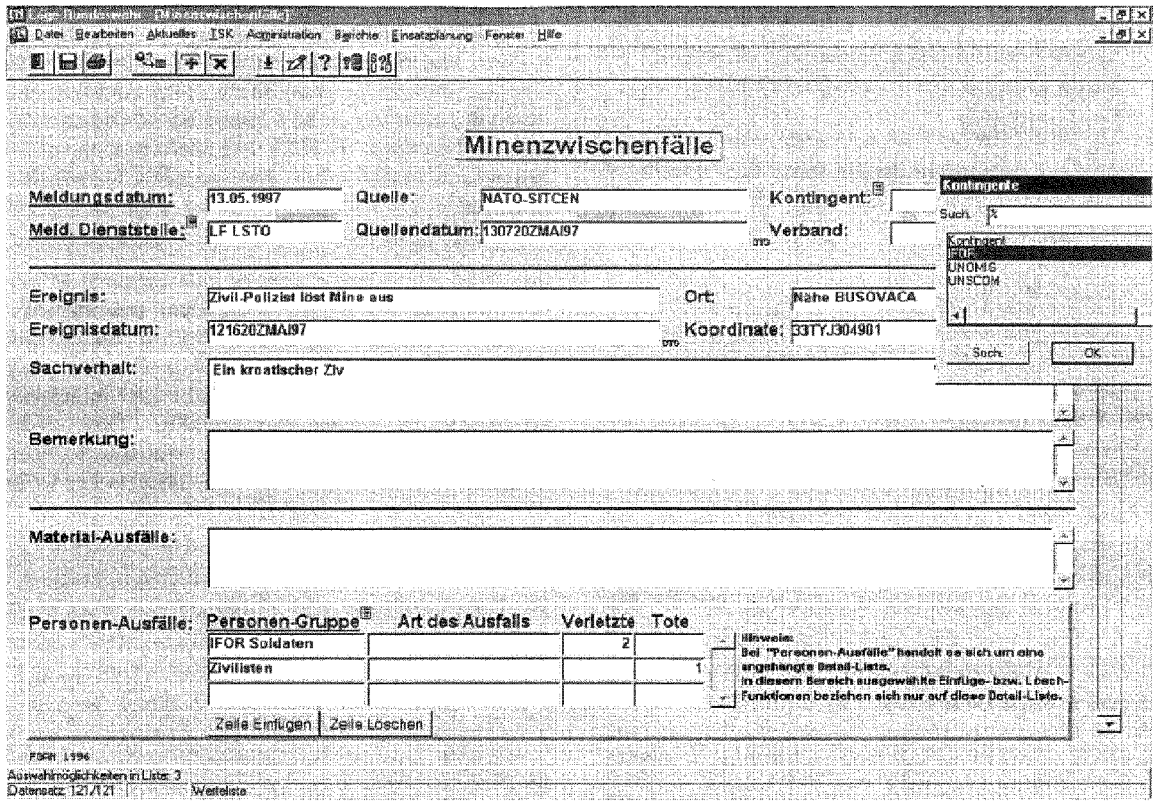

Figure 3: Scenario "Mine incidents" - input mask

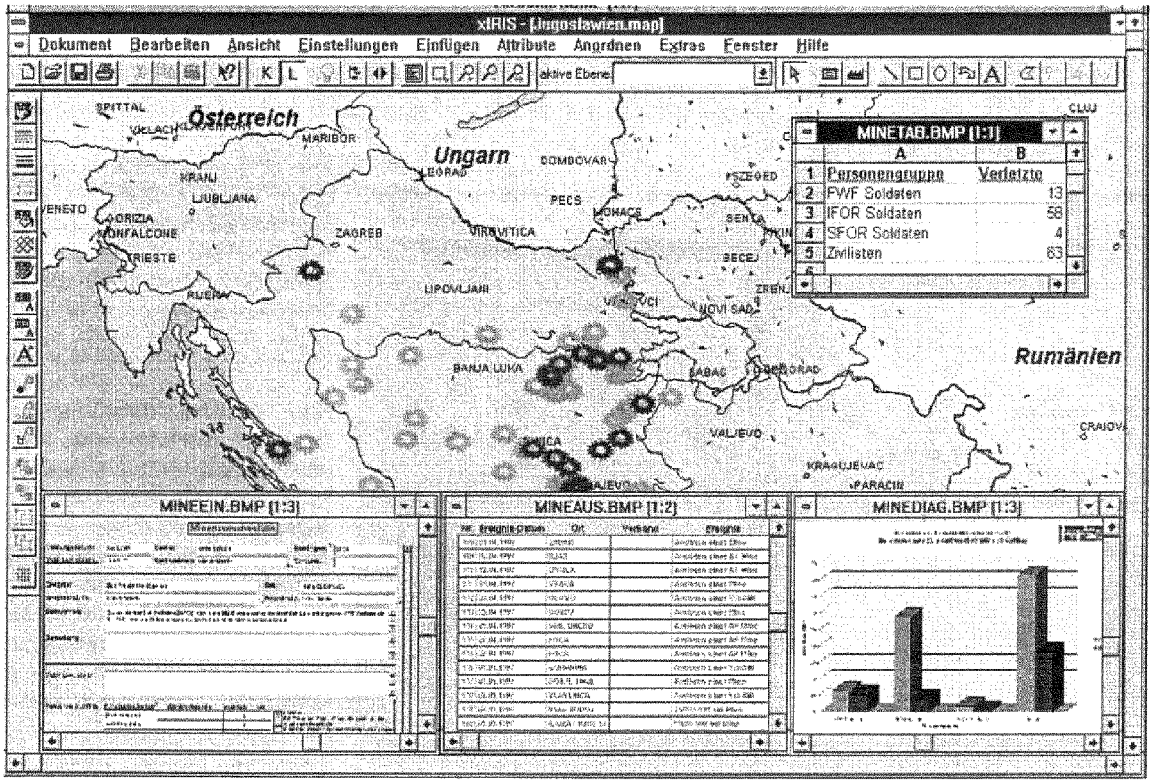

Figure 4: Scenario "Mine incidents" – different views

4 Rapid Application Development of CCIS based on CASE Technology

The advantage of components is their universality. However, it has to be taken into account that this flexibility may cause problems in the management of the total system. This is especially true in military CCIS of high complexity. Such systems comprehend a high quantity of data and often show a wide variety of heterogeneous data. In addition, there are various prerequisites according to the different user groups for the interaction with the data. There are requirements for inputting and manipulating as well as for analyzing, evaluating and presenting the data. There is, after all, the necessity for making important decisions in short time based on presented information. The aforementioned characteristics require especially that human factors principles have to be incorporated in the development of the user interface.

As mentioned before, a well defined framework needs to be defined, so that independent components can be embedded and cooperate as generic parts of a system. Within this framework, is may be necessary to modify components in a short time. Rapid Application Development (RAD) techniques allow an evolutionary implementation and optimization of components. This will be demonstrated for typical database applications.

The approach described herein deals with the application of integrated CASE technology (iCASE)[1] (Hoppe and Mempel 1998, Kurbel et al. 1994) in the development of ergonomically designed user interface components for a database of a CCIS (Kaster et.al. 1999). The technical platform is the Oracle Developer Suite.

4.1 CASE Technology - Bridging the Gap between User Requirements and Information Systems

The architecture of modern information systems is characterized by a decentral processing along with central administration of information. In order to use the data interactively within this client/server architecture consistent and transparent interfaces have to be developed. These enable task and situation dependent access, input and recherché of database information.

Figure 5 illustrates a general procedure to design a system using CASE technology. Based on the analysis of user requirements the application of CASE enables an almost fully automated and standardized creation of user forms for accessing the database. The application domain is structured in organizational items and processes (business process model). In the following system modeling phase information objects of the process model are represented in an entity relationship diagram (ERD) and functions are defined in a function hierarchy diagram (FHD). In using so-called wizards the application software is nearly automatically developed. The entities are transformed into tables of the database (database design) and the functions are transformed into the application design.

At this point available techniques of CASE were utilized to incorporate *ergonomic principles* in the design process of the application user interfaces. Oracle CASE offers the use of *libraries*, where the functionality of different forms is defined and stored in form of centralized, reusable and consistent code. It allows the definition of *preferences*, where general adjustments about layout, e.g. scrollbar position, take place. Furthermore, *templates* are used, e.g. to define the exact information about the layout of the header or footer or the button palette. Last but not least, reusable *master forms* can be defined, in order to avoid redundancy and to assure consistent behavior of several application forms in an object-oriented manner.

Using the description of the application domain in the repository and the predefined dialog components as well as presentation rules a consistent layout of the user interfaces is created. Integrated support functions enable user guidance and user assistance in operating the application forms, e.g. semi-automated filling of the masks, locking fields depending on user input, offering list of values for valid input as well as online help. The rapid application development supported by CASE allows an evolutionary development process and gives the chance for early user feedback.

4.2 Application of CASE

The heterogeneous data-sets in military command and control systems ask for various views on the data in order to satisfy user needs as well as to fulfill task requirements. Consequently, the visualization of the data has been conducted in various manners (Fig. 3 and 4).

(1) Using the CASE technology the data were visualized by several Oracle database tools. FORMS enables interaction in tabular form, GRAPHICS supports pictorial representation, and REPORTS generates written protocols.

(2) According to user requests the forms were connected with Microsoft Office tools. Forms with free text were coupled with WORD, tabular data were coupled with EXCEL, single documents can be transformed to PowerPoint in order to prepare presentations. In this manner users can work on their data in familiar environments.

(3) Another connection took place with a situation display program. Here, the forms are displayed as dialog components on digital maps, where interaction results in the modification of the situation represented on the map.

(4) In order to fulfill the requirements for an universal user interface the presentation of the database information can be presented based on Internet technology. The user can access the database via a commercial web browser.

[1] iCASE = integrated Computer Aided Systems Engineering

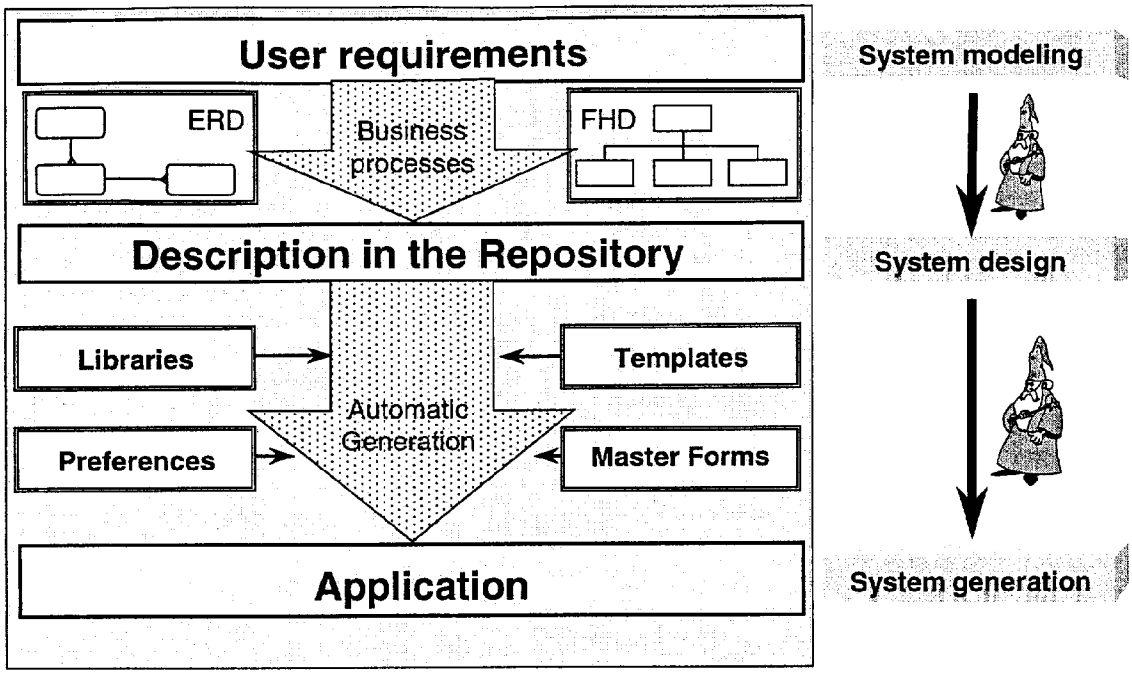

Figure 5: System design with iCASE technology enables the implicit incorporation of ergonomic principles

5 Conclusions

Current problems in the use of CCIS are often caused by insufficiencies of the human-machine interface. In order to overcome these problems it is necessary to take notice of strategies which help to create task oriented user interfaces that fulfill the requirements specified in ergonomic standards (ISO 1995, VDI 1990).

Componentware approaches provide means that support the practical realization of systems based on precise theoretical concepts. Design patterns help to structure the application domain by delivering ready-made design solutions.

Besides theoretical aspects, toolkits are required that support complex development processes. In this paper Rapid Application Development was referred to as a way that provides solutions in a short time. This method allows early user participation which is extremely important from a human factors point of view. The user interface for the access to the heterogeneous data can be created by using iCASE technology. Such a platform can provide the basic means to explicitly incorporate ergonomic knowledge in the design process.

The paper presented examples for benefits of componentware technology in a macro-view (considering a system as a whole) as well as in a micro-view (designing specific MOTS products).

In the research project it was demonstrated that the application of componentware allows the creation of interfaces in individual user environments in a consistent, complete and standard conform manner. The use of Internet technology will strengthen this effect.

6 References

Booch, G., Jacobson, I., Rumbaugh, J., The Unified Software Development Process, Addison-Wesley, 1999.

Fowler, M. Analysis Patters, Addison-Wesley, 1997.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

Griffel, F. Componentware: Konzepte und Techniken eines Softwareparadigmas. dpunkt-Verlag, 1998.

Hoppe, R. & Mempel, M. Oracle Designer R2.1, Addison-Wesley, 1998..

Kaster, A., Kaster, J., Kütttelwesch, H. und Meister, M. Ergonomische Systemgestaltung unter Anwendung der CASE-Technology. FKIE-Bericht Nr. 1, FGAN, Wachtberg. 1999.

Kurbel, K., Eicker, S., Kersten, F. Schnieder, T. and A. Teubner. I-CASE bei der Entwicklung eines großen Informationssystems: eine Information-Engineering-Fallstudie Wirtschaftsinformatik, 36, 2, 130-144. 1994.

Microsoft: Microsoft White Paper: DCOM Architecture, 1998.

Myers, B.A. User Interface Software Tools; ACM Transactions on Computer Human Interaction, vol.2 no. 1,pp. 64-103. 1995.

Object management Group (OMG): CORBA2.2 Specification: OMG98-2-33. 1998.

Plasil, F., Stal. M. An Architectural View of Distributed Objects and Components in CORBA, Java RMI, and COM/DCOM. Software Concepts & Tools, Springer. 1998.

Schrciber-Ehle, S. Adaptable GUIs Based on the Componentware Technique. In: Proceedings of HCI International, Munich 1999. Lawrence Erlbaum Associates.